A methodology and an implementation to perform live time-traversal queries on RDF datasets

ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Second Cycle Degree Programme in

Digital Humanities and Digital Knowledge

Final Dissertation in

Open Science

Supervisor Prof. Silvio Peroni

Presented by Arcangelo Massari

Co-supervisor Prof. Fabio Vitali

Session II

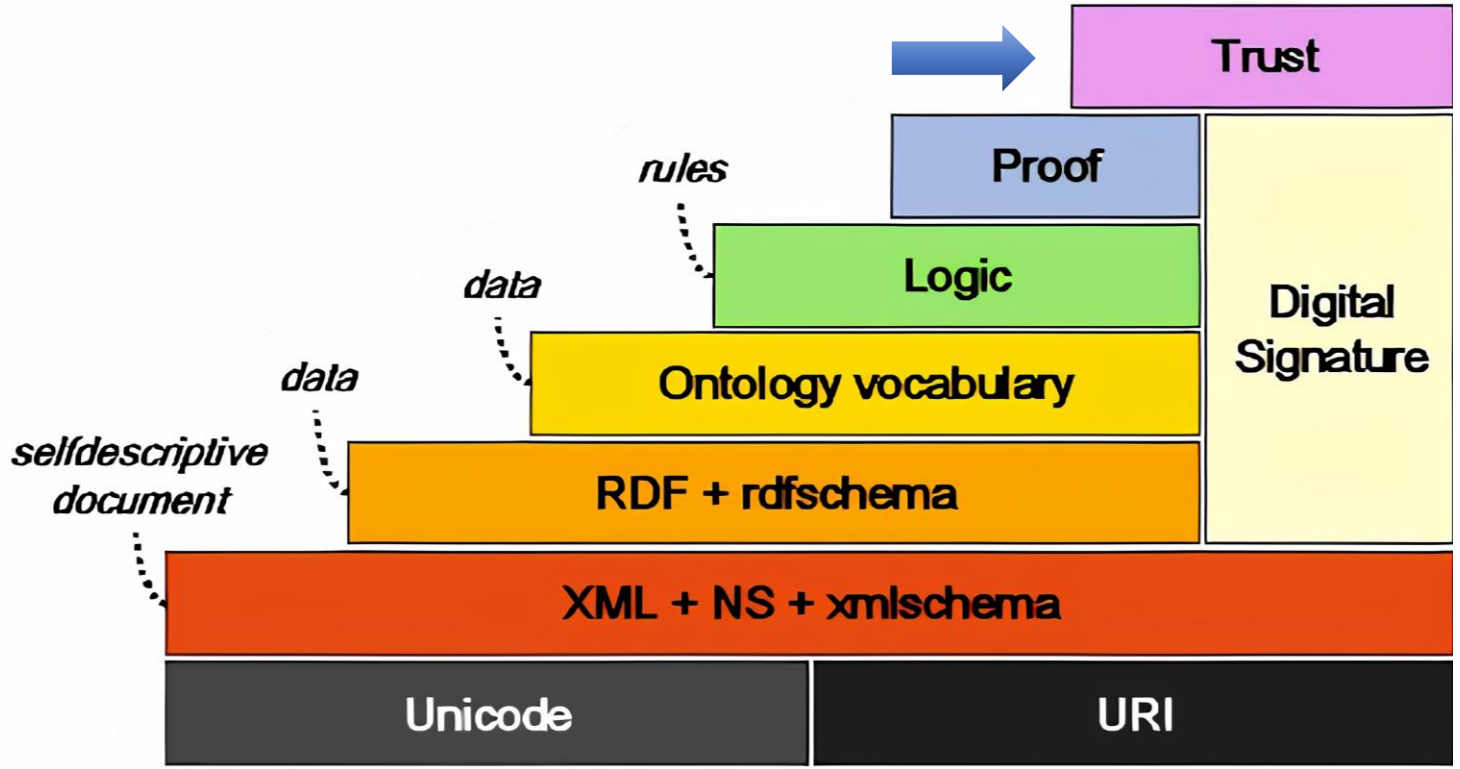
Academic Year

2020-2021

Verifiability, not truth

"The threshold for inclusion in Wikipedia is verifiability, not truth" (Garfinkel 2008)

- Trustworthiness must be evaluated by each application probing by the **context** of the statements (Koivunen and Miller 2001)
- **Provenance**: people, institutions, entities, and activities involved in producing, influencing, or delivering data

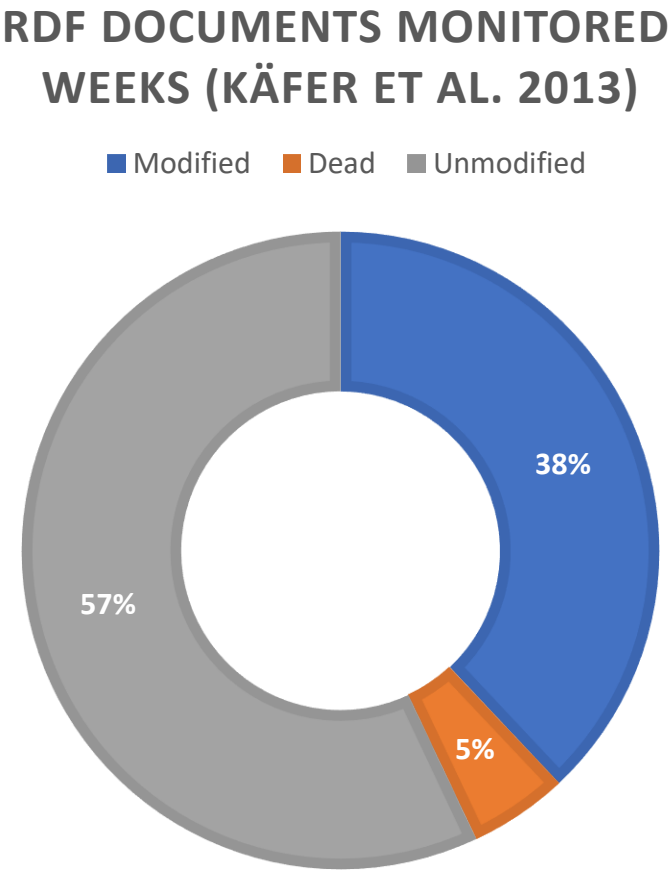


Garfinkel, S.L. 2008. 'Wikipedia and the Meaning of Truth'. *MIT Technology Review*

Koivunen, M.-R., and E. Miller. 2001. 'Semantic Web Activity'. W3C

Data evolves

- Natural evolution of **concepts**, related to a change of:
  - Place
  - Jurisdiction
  - Subjective perception of the receiver
- Correction of **mistakes**
- The latest version of knowledge may not be the most accurate



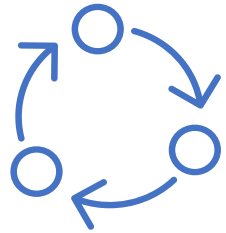
Käfer, Tobias, Ahmed Abdelrahman, Jürgen Umbrich, Patrick O'Byrne, and Aidan Hogan. 2013. 'Observing Linked Data Dynamics'. 7882:213–27. Lecture Notes in Computer Science. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-38288-8_15

Gap and objectives

Gap

The most extensive RDF datasets do **not** use RDF to track changes and provenance

Some of them, such as YAGO 4, record provenance but not changes



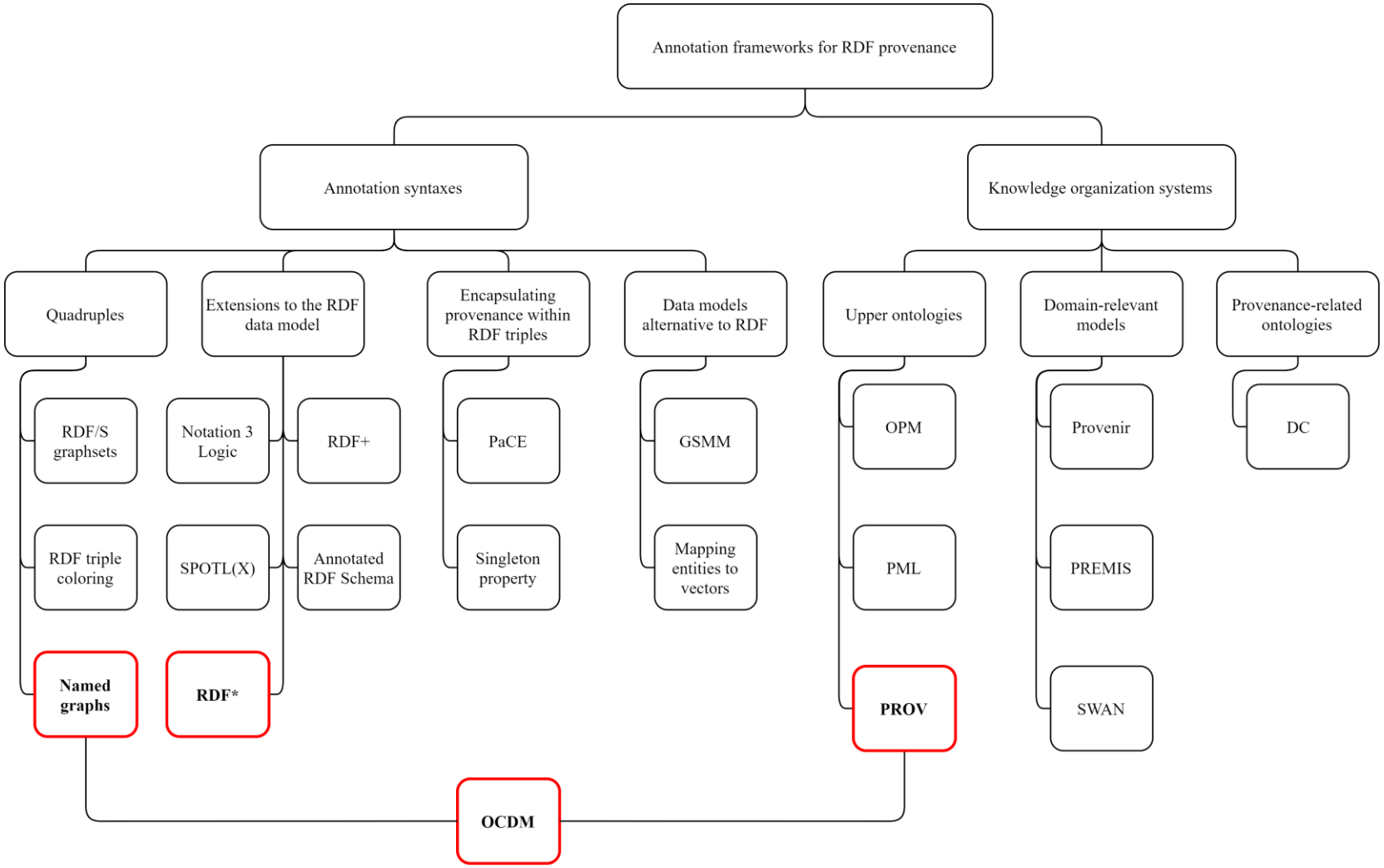
Objectives

Identify a provenance metadata model compliant with RDF

A methodology to perform live time-traversal queries on RDF datasets and software based on this procedure

Representing provenance in RDF

- Origin of the **first sin**: RDF 1.0 and **RDF Reification** (Sikos and Philp 2020)
- Many alternatives have no concrete implementation and are only **theoretical models**
- **Named Graphs** and the **Provenance Ontology** are the most adopted approaches
- Recent promising solution: **RDF***

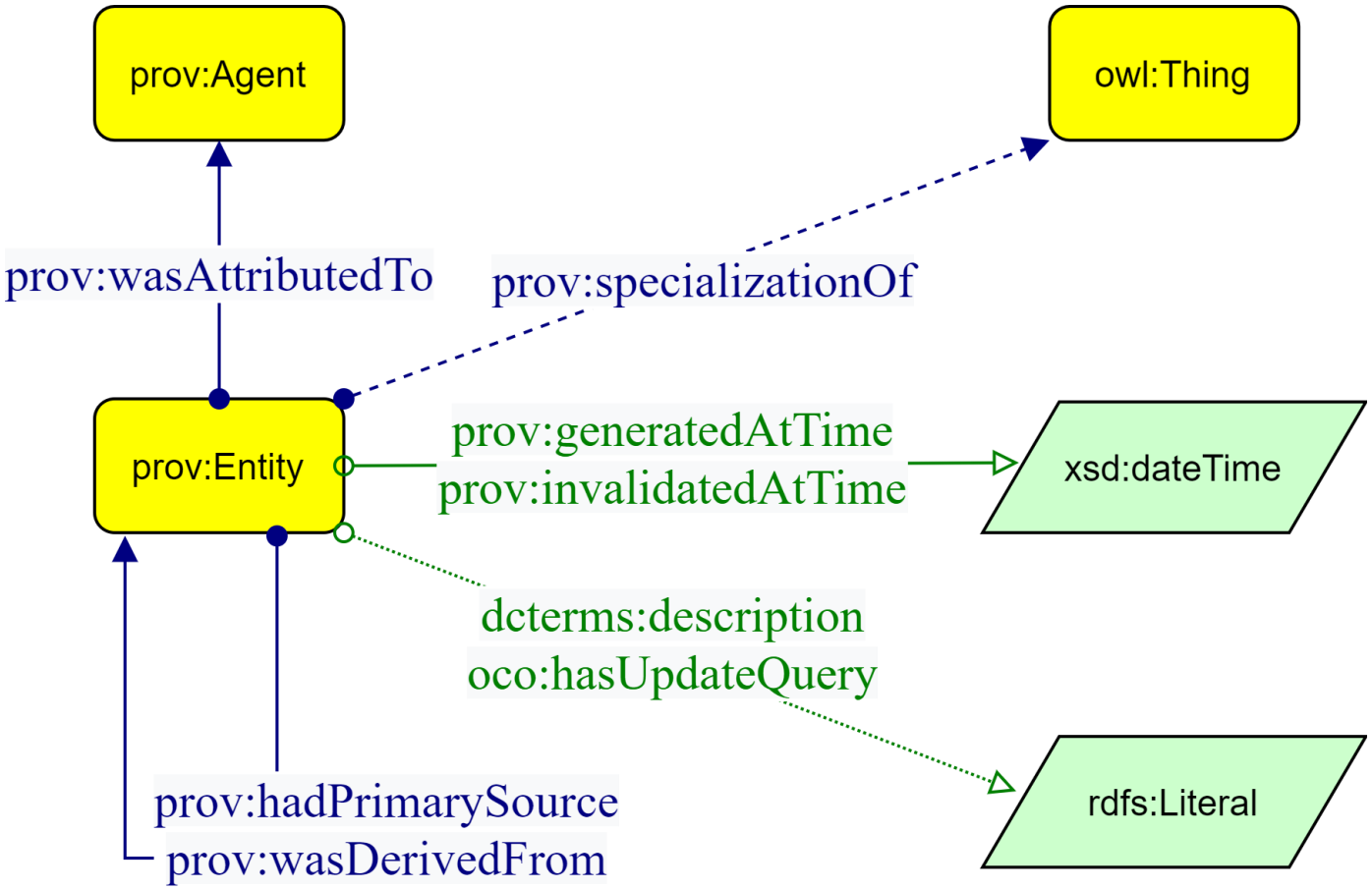


Sikos, Leslie F., and Dean Philp. 2020. 'Provenance-Aware Knowledge Representation: A Survey of Data Models and Contextualized Knowledge Graphs'. Data Science and Engineering 5 (3): 293–316. doi: 10.1007/s41019-020-00118-0

OpenCitations Data Model

- It combines **Named Graphs** and the **Provenance Ontology**
- It introduces a system to **simplify restoring** an entity's status at a given time
- It is **concretely used**
- It is **generic**

The methodology and software presented in this work leverage OCDM for all these reasons



Daquino, Marilena, Silvio Peroni, and David Shotton. 2020. 'The OpenCitations Data Model'. figshare

Methodology

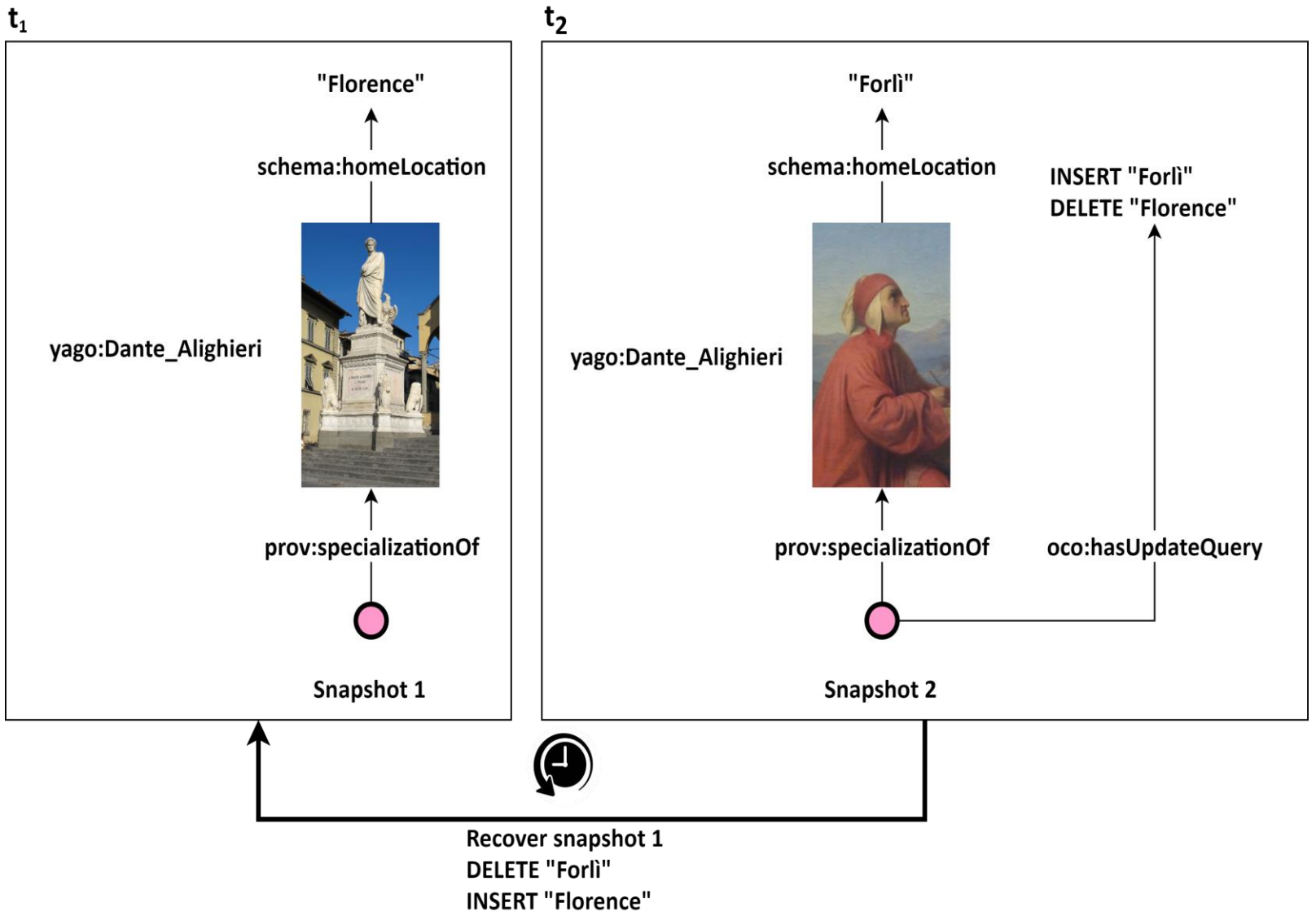
🎯 **Goal**: achieve all the time-related retrieval functionalities (Fernández, Polleres, and Umbrich 2015)

	Materialization	Structured query	
		Single-time ⏳	**Cross-time** ⏳⏳⏳
Version	Version materialization *Recovering an entity state at a given time*	Single-version structured query *Resolving a SPARQL query on a specific entity's snapshot*	Cross-version structured query *Resolving a SPARQL query on all the dataset's versions*
Delta Δ	Delta materialization *Recovering the difference between two states*	Single-delta structured query *Resolving a SPARQL query on deltas within a certain period*	Cross-delta structured query *Resolving a SPARQL query on all the dataset's deltas*

Fernández, J.D., A. Polleres, and J. Umbrich. 2015. 'Towards Efficient Archiving of Dynamic Linked'. In DIACRON@ESWC, 34–49. Portorož, Slovenia: Computer Science

Methodology: version materialization

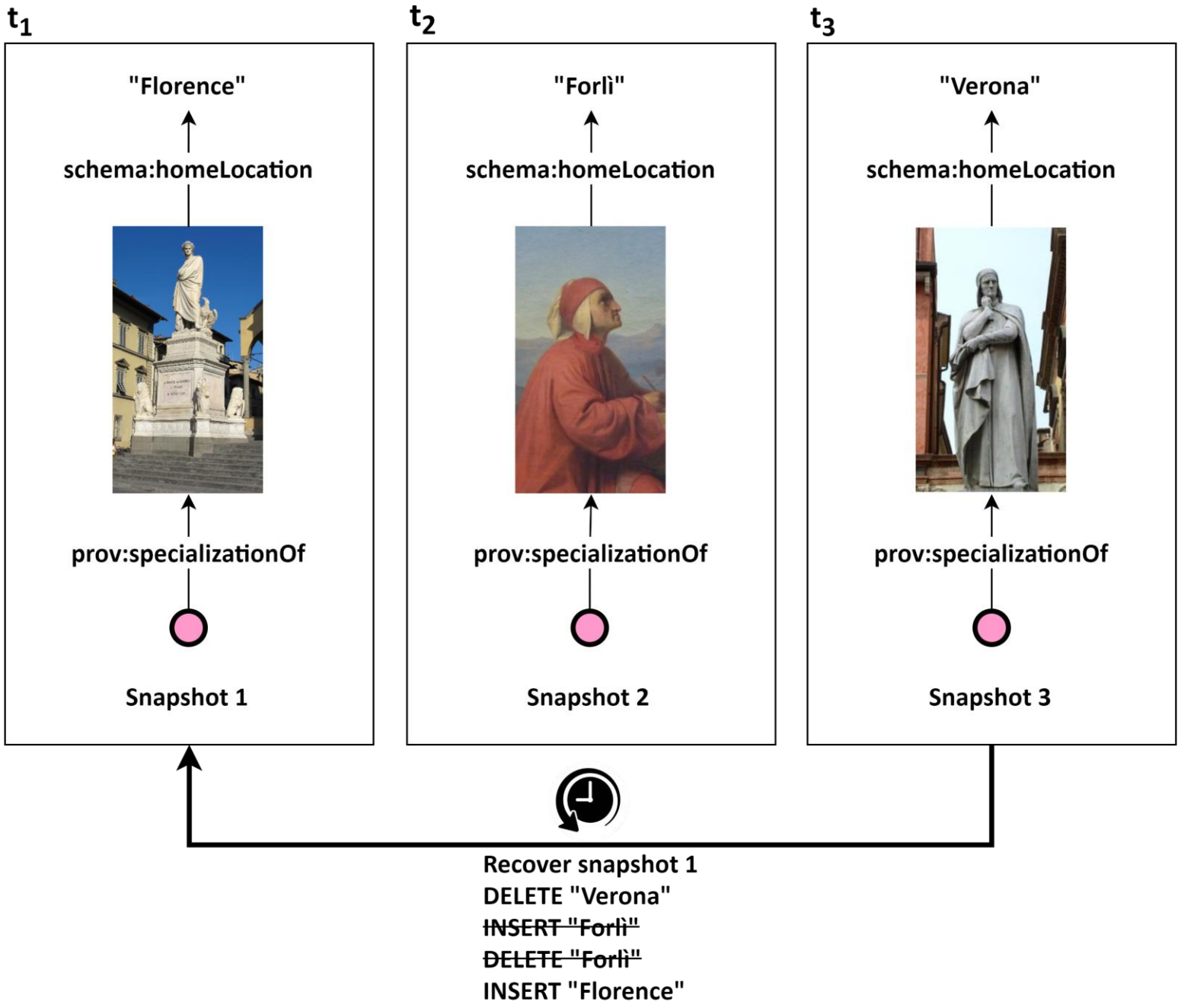
- **Materialization** means returning an entity **state** at a given **time**
- To recover the status of an entity to a particular snapshot s_i, apply the **inverse update queries** from the most recent snapshot s_n to s_{i+1}



Key message: what was deleted must be inserted, and what was inserted must be deleted to rewind an entity's time

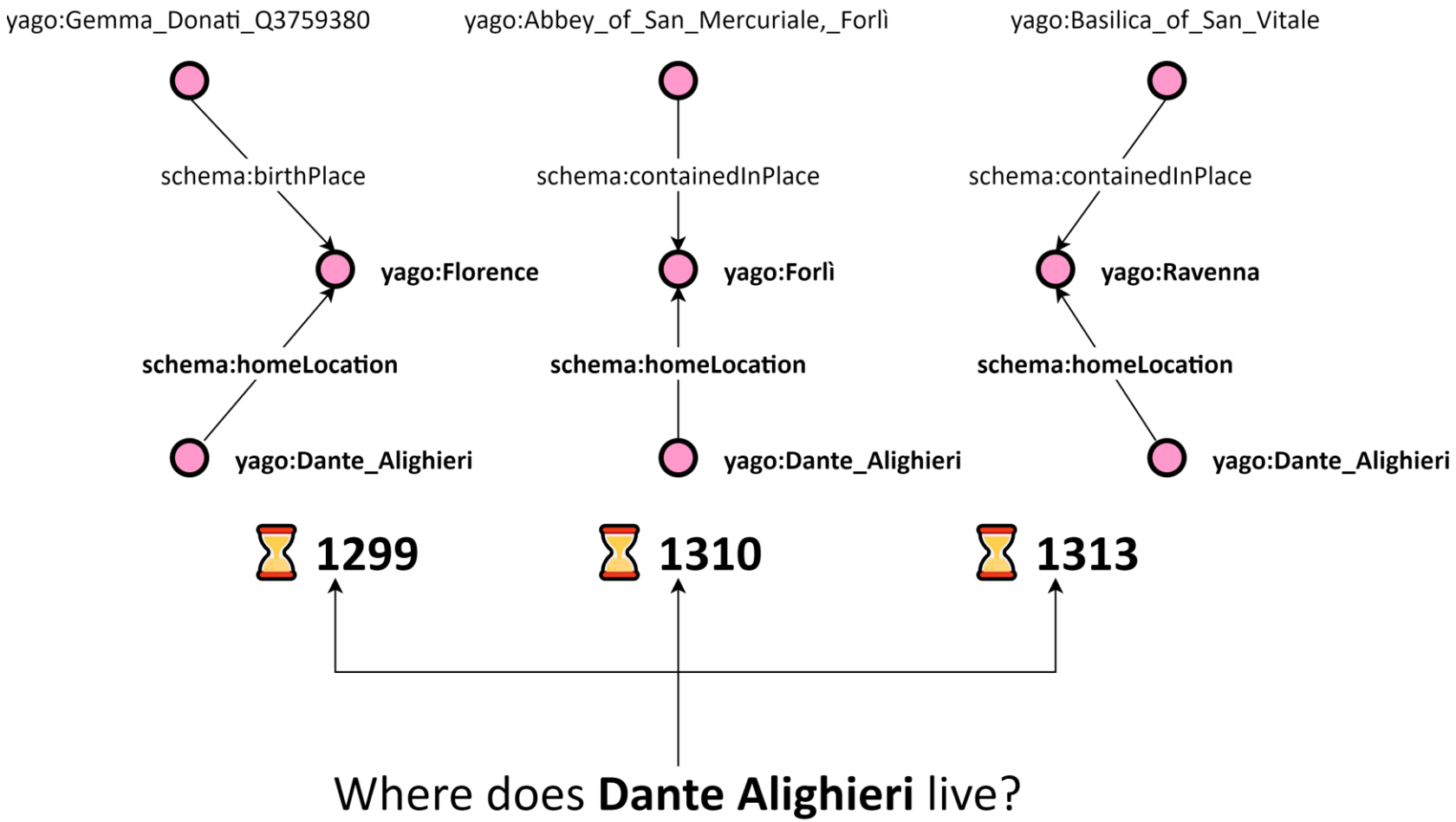
Matarialization: time leap

- ⚠ **Problem**: How to get the status of a resource at a given time without reconstructing the whole history?
- 🎯 **Goal**: making a time leap from the present state to the state of interest
- 💡 **Solution**: $s_{n-2} = -\Delta s_n + -\Delta s_{n-1}$



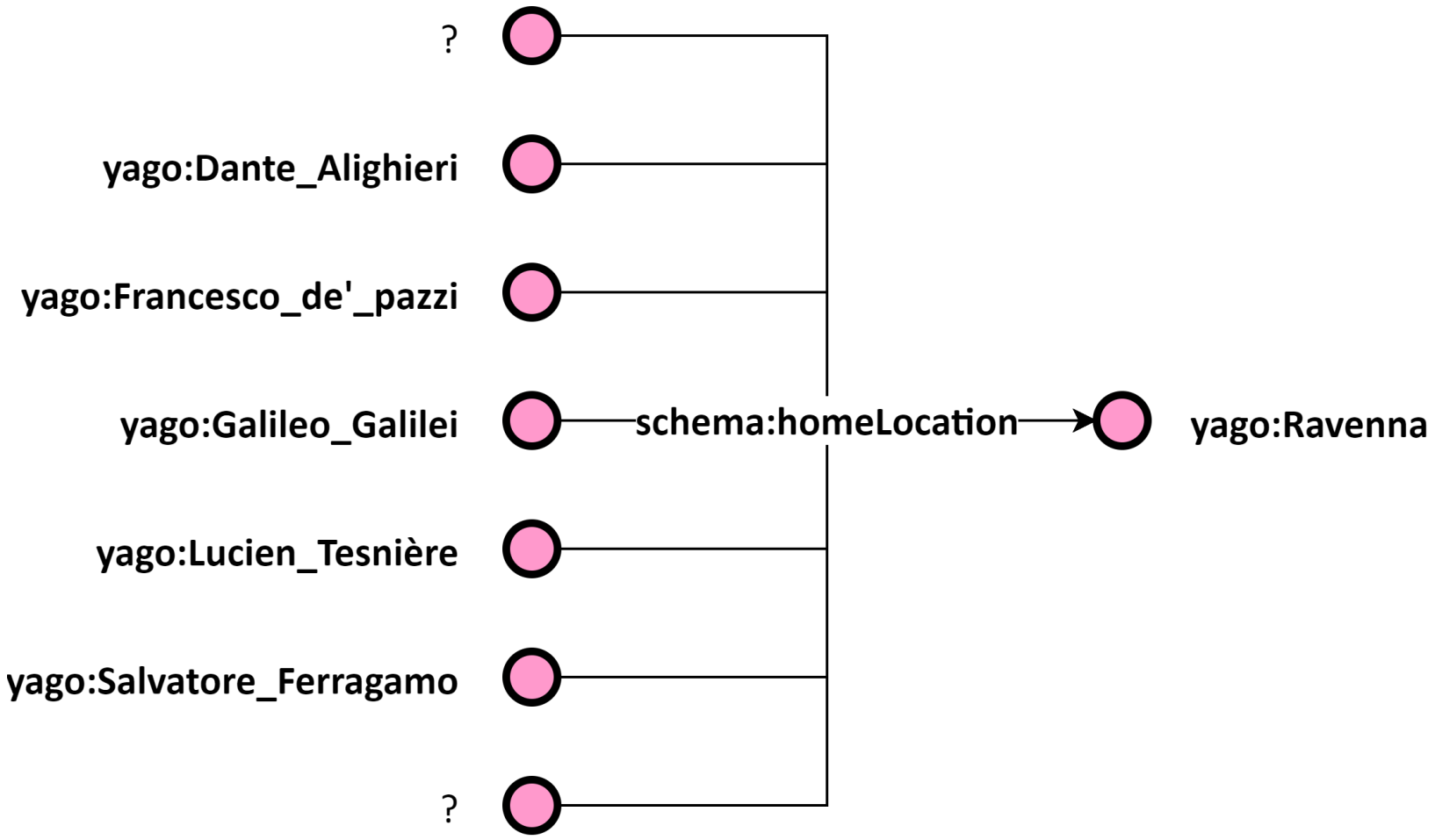
Methodology: version structured query

- **Version structured query**: resolves a SPARQL query on a **specific** entity's **snapshot** or on **all** the dataset's **versions**
- ⚠ **Problem**: restoring as many versions as snapshots would generate **massive** amounts of data
- 💡 **Solution**: reconstruct only the past resources **significant** for the user's query



Time-traversal query with unknown subjects

- ⚠ **Problem**: If the **subject** is **unknown**, answering a time-agnostic query is more **demanding**
- All entities that have ever been related to known predicates or objects must be identified and reconstructed
- 💡 **Solution**: this information can be found in the saved **delta strings**



Who **lives** in **Ravenna**?

Methodology: single-delta and cross-delta structured query

- **Single-delta structured query**: resolves a SPARQL query on **deltas** within a certain **period**
- **Cross-delta structured query**: resolves a SPARQL query on **all** the dataset's **deltas**
- 🎯 **Purpose**: understanding which resources have **changed** to narrow the field and achieve faster queries on versions
- 🗲 **Faster** than a time-travel-query: no need to rebuild relevant entities' pasts since deltas are explicitly stored according to the OpenCitations Data Model

Time-Agnostic Library

- A **Python** ≥ 3.7 library
- Available **open-source** on GitHub under the ISC License
- Distributed as a package and can be installed with **pip** via a terminal command
- **Test-Driven Development** (TDD) was adopted as a software development process
  - **72** tests

```
# MATERIALIZATION
agnostic_entity = AgnosticEntity(res=RES_URI, config_path=CONFIG_PATH)
output = agnostic_entity.get_state_at_time(
        time=(START, END), include_prov_metadata=BOOL)

# VERSION QUERY
agnostic_query = VersionQuery(
        query=QUERY_STRING,
        on_time=(START, END),
        config_path=CONFIG_PATH)
output = agnostic_query.run_agnostic_query()

# DELTA QUERY
agnostic_entity = DeltaQuery(
        query=QUERY_STRING,
        on_time=(START, END),
        changed_properties=PROPERTIES_SET,
        config_path=CONFIG_PATH)
agnostic_entity.run_agnostic_query()
```

Benchmarks: time & RAM

- Two benchmarks: **execution times** and **RAM**.
- **Ten** different **use cases**
- **Ten iterations** to avoid outliers
- **Efficient** for:
  - Any **materialization**
  - Structured queries with **known subjects**
- **Inefficient** for:
  - SPARQL queries with **unknown subjects**

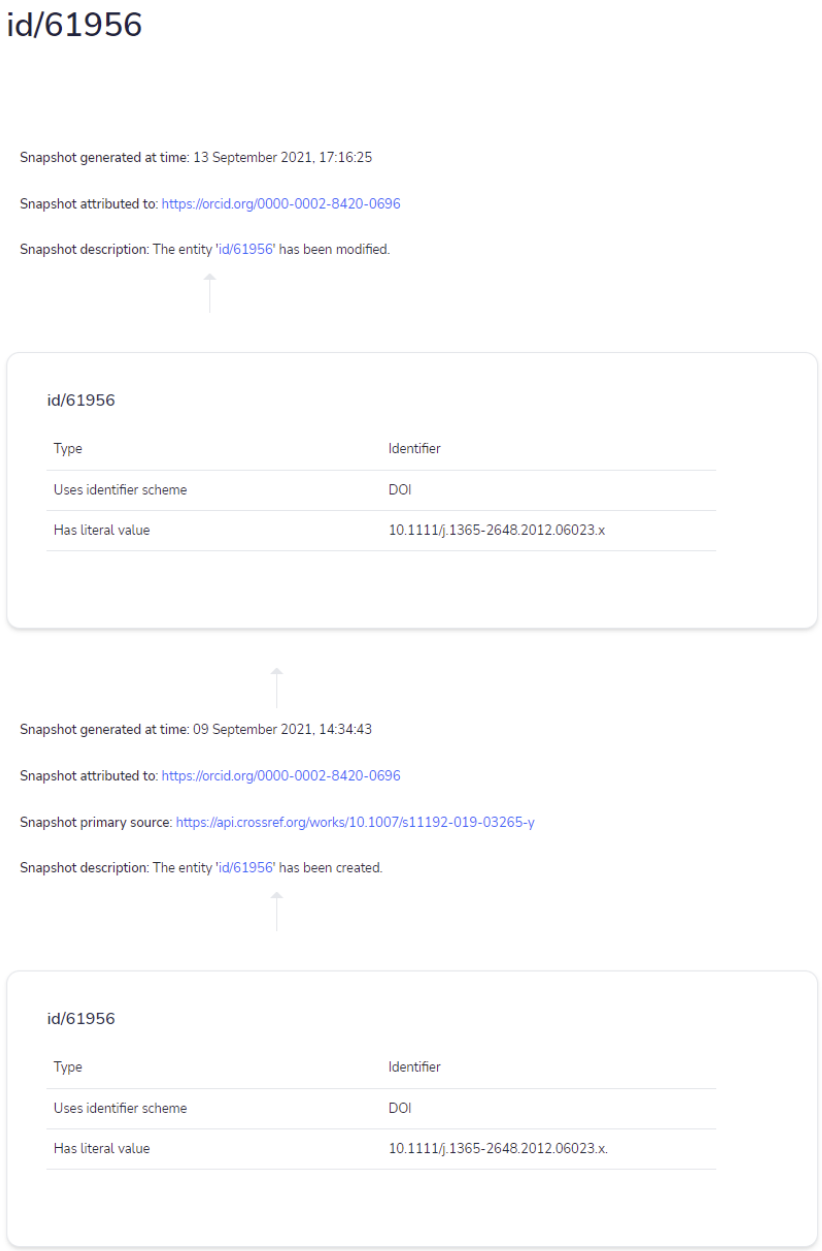
Retrieval functionality	Median time (s)	Median RAM (MB)
Materialization of all versions	0.583s	51.5 MB
Cross-version structured query	0.604s	51.7 MB
Cross-version structured query where only the predicate and object are known	581s	519 MB

Value and originality

Software	Version materialization	Delta materialization	Single-version structured query	Cross-version structured query	Single-delta structured query	Cross-delta structured query	Live
PromptDiff	+	+	-	-	-	-	+
SemVersion	+	+	-	-	-	-	+
(Im, Lee, & Kim, 2012)	+	+	+	-	+	+	-
R&Wbase	+	+	+	-	-	-	+
x-RDF-3X	+	-	+	+	-	-	-
v-RDFCSA	+	+	+	+	+	+	-
OSTRICH	+	+	+	-	-	-	-
(Tanon & Suchanek, 2019)	+	+	+	+	+	+	-
time-agnostic-library	**+**	**+**	**+**	**+**	**+**	**+**	**+**

Time-Agnostic Browser

- It enables recovering all the past of an entity from its URI and performing time-traversal queries through a **graphical user interface**
- Client-side: Javascript, Jinja2, **Vue.js**
- Server-side: **Flask**
- http://51.15.198.219/



Time Agnostic Browser

Explore | Query

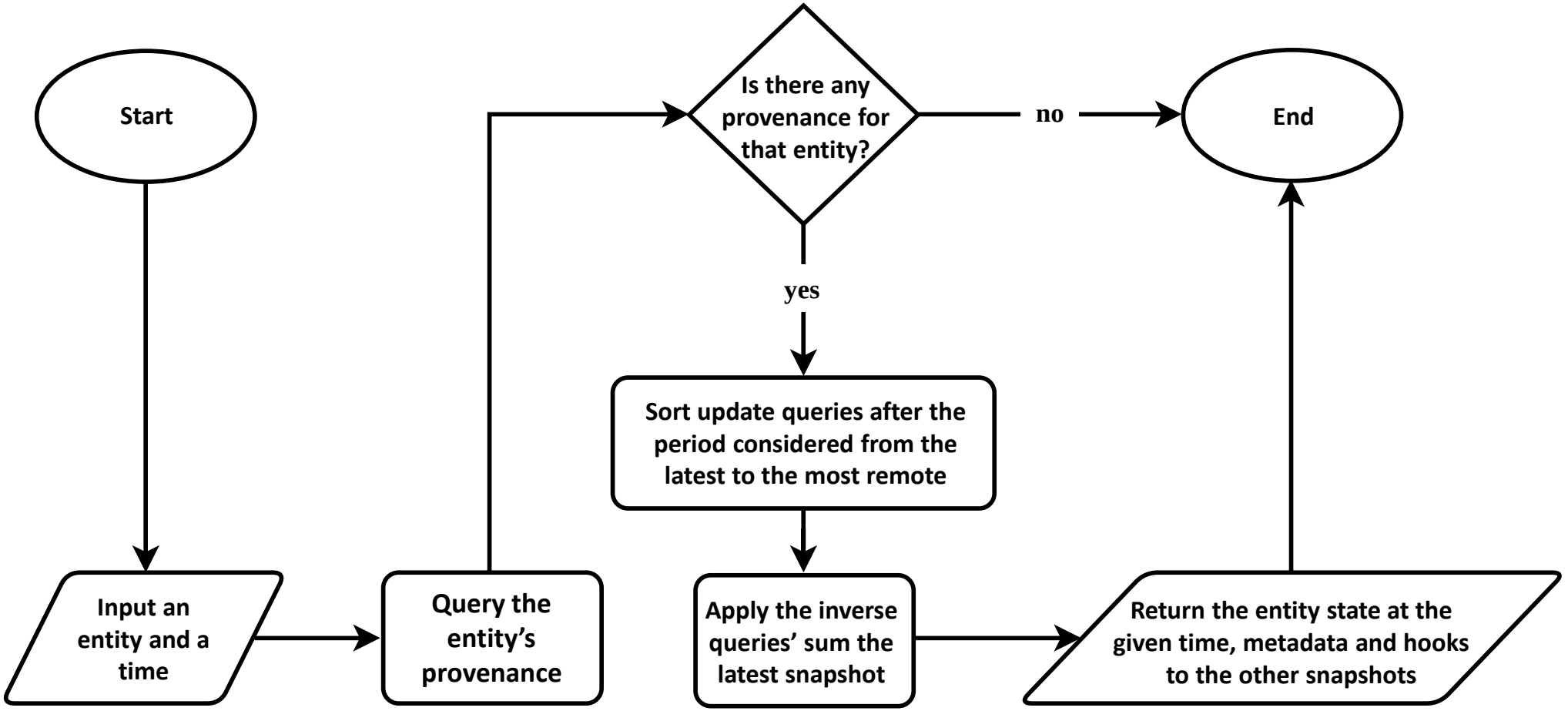
Enter a URI to begin navigation

https://github.com/opencitations/time-agnostic-library/br/86766

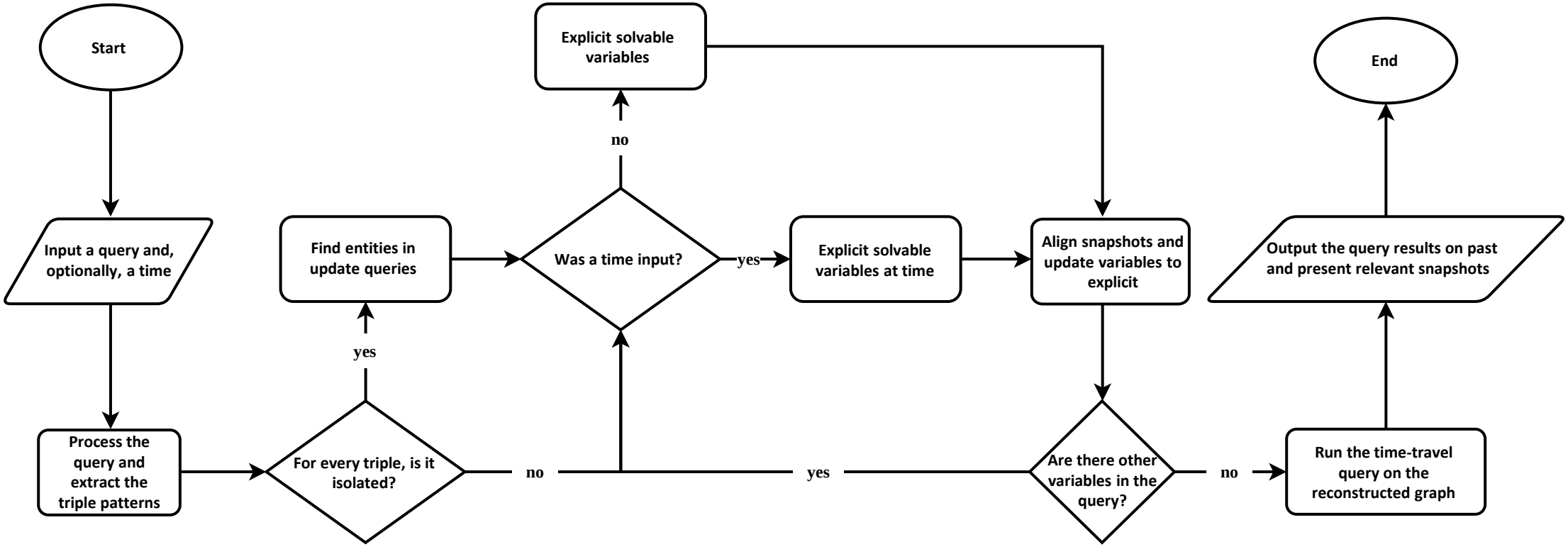
Submit the query

The End

Materialization: flowchart



Methodology: single-version and cross-version structured query



Triple patterns: joined and isolated

- **Joined triple pattern**: a path exists between its subject variable and a subject URI in the query.
  - Explicate the variables **recursively.**
- **Isolated triple pattern**: it is wholly disconnected from the other patterns in the query, and its subject is a variable.
  - Search for relevant entities in relevant **update queries**.

```
PREFIX literal: <http://www.essepuntato.it/2010/06/literalreification/>
PREFIX cito: <http://purl.org/spar/cito/>
PREFIX datacite: <http://purl.org/spar/datacite/>
SELECT DISTINCT ?br ?id ?value
WHERE {
    <https://github.com/opencitations/time-agnostic-library/br/86766> cito:cites ?br.
    ?br datacite:hasIdentifier ?id.
    ?id literal:hasLiteralValue ?value.
}
```

Example of a SPARQL query including only joined triple patterns.

```
PREFIX literal: <http://www.essepuntato.it/2010/06/literalreification/>
SELECT ?literal
WHERE {
    ?id literal:hasLiteralValue ?literal.
    FILTER REGEX(?literal, "\.$")
}
```

Example of a SPARQL query including an isolated triple pattern.

Methodology: single-version and cross-version structured query